DATA TYPES

Declaring and Initializing Variables,
Type Conversion

Designed by Parul Khurana, LIECA.

Data Types

- C supports several types of data, each of which may be represented differently within the computer's memory.

Basic Data Types

Type	Length	Description	Range
int	16 bits	integer quantity	-32,768 to 32,767
char	8 bits	single character	-128 to 127
float	32 bits	floating-point number	3.4 * (10**-38) to 3.4 * (10**+38)
double	64 bits	double-precision floating-point number	1.7 * (10**-308) to 1.7 * (10**+308)

- The basic data types can be augmented by the use of the data type qualifiers short, long, signed and unsigned.

Basic Data Types with Qualifiers

Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 * (10**-38) to 3.4 * (10**+38)
double	64 bits	1.7 * (10**-308) to 1.7 * (10**+308)
long double	80 bits	3.4 * (10**-4932) to 1.1 * (10**+4932)

Declaring Variables

- A declaration associates a group of variables with a specified data type. All variables must be declared before they can appear in executable statements.

- A declaration consists of a data type, followed by one or more variable names, ending with a semicolon.

Examples Declaring Variables

- A C program contains the following type declarations:

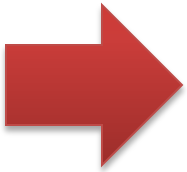
```
int a, b, c;
float root1, root2;
char flag;
```

- Thus , a, b and c are declared to be integer variables, root1 and root2 are floating-point variables and flag is a char-type variable.

Examples Declaring Variables

- *These declarations could also have been written as follows:*

```
int a, b, c;
float root1, root2;
char flag;
```



```
int a;
int b;
int c;
```

```
float root1;
float root2;
```

```
char flag;
```

- This form may be useful if each variable is to be accompanied by a comment explaining its purpose. In small programs, however, items of same type are usually combined in a single declaration.

Initializing Variables

- Initial values can be assigned to variables within a type declaration.

- To do so, the declaration must consist of a data type, followed by a variable name, an equal sign(=) and a constant of the appropriate type.
- *A semicolon must appear at the end, as usual.*

Examples Initializing Variables

- *A C program contains the following type declarations:*

```
int a = 12;
int b = 2400;
int c = -320;
```

```
float root1 = 3.12;
float root2 = -8.2;
```

```
char flag = '*';
```

- *Thus, a, b and c are integer variables whose initial values are 12, 2400 and -320, root1 and root2 are floating-point variables whose initial values are 3.12 and -8.2 and flag is a char type variable initially assigned the character '*'.*

Type Conversion

- When variables of one type are mixed with variables of another type, a type-conversion will occur.

- In an assignment, the type conversion rule is easy:
  - The value of the right side(expression side) of the assignment is converted to the type of the left side (target variable), as illustrated here:

Example Type Conversion

```
int x;
char ch;
float f;
```

```
ch = x;     // line 1
x = f;      // line 2
f = ch;     // line 3
f = x       // line 4
```

- In line 1, the left high order bits of the integer variable x are lopped off, leaving ch with the lower 8 bits. If x were between 255 and 0, ch and x would have identical values.
- Otherwise, the value of ch would reflect only the lower-order of bits of x.

Example Type Conversion

```
int x;
char ch;
float f;
```

```
ch = x;   // line 1
x = f;    // line 2
f = ch;   // line 3
f = x     // line 4
```

- In line 2, x will receive the non-fractional part of f.
- In line 3, f will convert the 8-bit integer value stored in ch to the same value in the floating-point format.

Example Type Conversion

```
int x;
char ch;
float f;
```

```
ch = x;   // line 1
x = f;    // line 2
f = ch;   // line 3
f = x     // line 4
```

- In line 4, except that f will convert an integer value into floating-point format.

Example Type Conversion

- When converting from integers to characters and long integers to integers, the appropriate amount of high order bits will be removed.

- In many 16-bit environments, this means that 8 bits will be lost when going from an integer to a character and 16-bits will be lost when going from a long integer to an integer.

Example Type Conversion

- For 32-bit environments, 24 bits will be lost when converting from an integer to a character.

- *Remember that the conversion of an int to a float, or a float to a double, and so on, does not add any precision or accuracy.*

- *These kind of conversions only change the form in which value is represented.*

Summary Type Conversion

Target type	Expression type	Possible info loss
signed char	char	If value > 127, target is negative
char	short int	High-order 8 bits
char	int (16 bits)	High-order 8 bits
char	int (32 bits)	High-order 24 bits
char	long int	High-order 24 bits
short int	int (16 bits)	None
short int	int (32 bits)	High-order 16 bits
int (16 bits)	long int	High-order 16 bits
int (32 bits)	long int	None
int	float	Fractional part and possibly more
float	double	Precision, result rounded
double	long double	Precision, result rounded

The outcome of Common Type Conversions

Practice Questions

- Give suitable declarations for the variables to hold the values for :-
  - Your name.
  - Your age (can never be negative).
  - Your %age marks.
  - Your 10-digit mobile number.
  - Your Country represented by either I or O.
  - Boolean values 0 or 1.
  - To store the values that begin with 0x.